CSE 322: Computer Network Sessional
Data Link Layer

Level: 3, Term: 2 (January 2020)

In this assignment, you will implement a data link layer protocol to simulate flow control and error control mechanisms at the data link layer.

Task 1: Flow Control

Implement 1-bit sliding window protocol. Suppose you have two DLL clients A and B who are communicating with each other. Since the sliding window protocol is a bidirectional protocol, **both A and B can be the sender of data frames**. You have to implement the protocol for the entities A and B.

The unit of data passed between the upper layer (layer 3 or network layer) and your protocol is a *packet*. Your sending entity, say A, will thus receive data from layer 3; your receiving entity, B should deliver correctly received data to layer 3 at the receiving side. The unit of data passed between A and B is the *frame*, which contains the *packet* as its payload.

Frame formation:
Whenever a sending entity, say A receives a packet from the network layer (layer 3), it creates a frame as follows and sends the frame to B (receiving entity) through the physical layer (layer 1).

| type of frame (data/ack)| seqNo | ackNo | payload | checksum | where,
a. each of the fields: type, seqNo, ackNo and checksum can be integers.
Set type = 0 for a data frame, type = 1 for an acknowledgement (ACK/NACK) frame, and type = 2 for a data frame with piggybacked acknowledgement.
b. payload field is of variable size determined by the size of the 'packet' as given by the network layer. For this assignment, keep payload size at 4 bytes.

Acknowledgement management:
After receiving a frame from A, B sends an acknowledgement frame (frame without any payload) if B does not have any outstanding data frame to send to A. Otherwise, B sends a **piggybacked acknowledgement** in the data frame it sends to A.

Timer-based retransmission:
Both data and acknowledgement frames may require timer-based retransmission if not received and acknowledged by the receiver of the frame. Please follow the guidelines of the previous assignment.

Task 2: Error Control

You will implement the error detection mechanism using CRC (Cyclic Redundancy Check) checksum. The generator polynomial should be taken as input in your code. Your code must print the input bit string to the CRC algorithm, generator polynomial, and the calculated CRC while computing CRC on the sender's side. On the receiver side, print the input bit string to the CRC algorithm, generator polynomial and an error message if a data transmission error has occurred.

Implementation Guideline

1. Adopt the code from your TCP assignment. Replace 'struct msg' with 'struct pkt'. Also replace 'struct pkt' with 'struct frm'. In each frame, keep an additional variable to designate the type of the frame.
2. Make necessary changes in the procedure names to replace the word 'layer5' with 'layer3' and 'layer3' with 'layer1'. For example, tolayer3(calling_entity,packet) will be renamed as tolayer1(calling_entity,frame); tolayer5(calling_entity,message) will be renamed as tolayer3(calling_entity,packet).
3. Unidirectional /bidirectional data transfer: Please follow the TCP assignment guideline to achieve this.
4. Implementation of piggybacked acknowledgement:
   To achieve piggybacking with minimal change in your code, we have come up with the following *protocol* which differs from the ideal implementation of piggybacking.

   After B receives a data frame *Fab* from A, B does not send an acknowledgement immediately. B sets a state variable called "OutstandingACK" to 1. If B receives a packet from its upper layer (layer 3) while OutstandingACK = 1, then B creates a data-ack frame *Fba* and piggybacks the acknowledgement of the previously received frame *Fab* in this frame. Otherwise, B creates a data frame.

   However, it is possible that B does not receive any data frame to piggyback the acknowledgement. In this case, A will eventually resend frame *Fab.* Upon receiving a duplicate frame, B will send the outstanding acknowledge frame.

   Example of piggybacked acknowledgement with 0 packet loss and 0 corruption:
   A receives a packet_A1 from layer3
   A sends frame_A1 to B
   Type = 0, Seq = 0, Ack = <don't care>, data = <content of packet_A1>, checksum =<calculated value>

   B receives a data frame from layer1
   No error has occurred

B waits to send Ack to A

// Either of the following two events can happen now, colored with green and blue
B receives a packet_B1 from layer3
B sends frame_B1 to A
Type = 2, Seq = 0, Ack = 0, data = <content of packet_B1>, checksum =<calculated value>

A times out
A resends frame_A1 to B
Type = 0, Seq = 0, Ack = <don't care>, data = <content of packet_A1>, checksum =<calculated value>

//The counter events of the above two events are as follows
A receives a data-ack frame from layer1
No error has occurred
A received Ack for frame_A1
A waits to send Ack to B

B receives a data frame from layer1
No error has occurred
Duplicate frame has been received
B sends Ack frame to A
Type = 1, Seq = <don't care>, Ack = 0, data = null, checksum =<calculated value>

……..

Test Cases

You may take 3 additional inputs in your code.
CRC steps: 0 -> do not show steps of CRC, 1 -> show steps of CRC
Piggybacking: 0 -> No piggybacked acknowledgement, 1->piggybacked acknowledgement
Generator polynomial: <define as you like>
Generate 3 separate output files for the following 3 cases. Each case carries 10 points.

Case 1: Test bidirectional data passing
Run the protocol by enabling bidirectional message passing and set
CRC steps = 0
Piggybacking = 0
Packet loss probability = 0.2
Corruption probability = 0.3

Number of message/packets = 5
Interval = 500

Case 2: Test bidirectional data passing with piggybacking
Run the protocol by enabling bidirectional message passing and set
CRC steps = 0
PiggyBacking = 1
Packet loss probability = 0.2
Corruption probability = 0.3
Number of message/packets = 5
Interval = 100

You may reduce the interval to produce at least one piggybacking case in your output.

Case 3: Test CRC algorithm implementation
Run the protocol by setting
Sending Entity = 0
CRC steps = 1
PiggyBacking = 0
Packet loss probability = 0
Corruption probability = 0.5
Number of message/packets = 2
Interval = 500

Submission Guidelines

Create a folder, use your student id as the folder's name, e.g., 1605xyz and copy all the files there. Only submit the source (.c) files and output (.doc) files. Then zip the folder and upload to Moodle in the assignment submission link that will be opened before the submission deadline. The deadline of submission for all sections is **December 05, 2020 at 2:00 pm.**

Useful links

Computing CRC: https://www.youtube.com/watch?v=A9g6rTMblz4
https://www.youtube.com/watch?v=wQGwfBS3gpk